IMPLICANT BASED SOLVER FOR XOR BOOLEAN LINEAR SYSTEMS

Jayashree Katti
Department of Information Technology
Pimpri Chinchwad College of Engineering
Pune,India
jayashree.katti@gmail.com

Virendra Sule
Department of Electrical Engineering
Indian Institute of Technology
Bombay, India
vrs@ee.iitb.ac.in

B.K.Lande
Vasantdada Patil College of Engineering
Bombay,India
bklande@gmail.com

ABSTRACT

An approach is presented for solving linear systems of equations over the Boolean algebra $B_0 = \{0,1\}$ based on implicants of Boolean functions. The approach solves for all implicant terms which represent all solutions of the system. Traditional approach to solving such linear systems is to consider them over the field $GF(2)$ and solve either by Gaussian elimination or Lanczos methods. One of the unfinished problems in Computer Science is that of developing scalable parallel solvers for such systems. The proposed approach based on implicants has inherent parallel structure for computation in terms of independent threads. We show that for sparse systems with a fixed bound on number of variables in any equation and using sufficient parallel resource, this approach requires $O(n)$ time where n is the number of variables. Hence this approach is expected to provide a scalable solution to the problem of solving large Boolean linear systems over large number of processors.

Keywords: XORSAT, Implicants.

1 INTRODUCTION AND MOTIVATION

Boolean Satisfiability problems arise in many applications such as cryptology, hardware and software verification, reliability, artificial intelligence, decision under logic constraints, computational studies of Biological networks (Crama and Hammer 2011), (Bolouri 2008) ,(Alan Veliz-Cuba and Laubenbacher 2014). In Computer Science, the problem of deciding satisfiability of Boolean formulas in Conjunctive Normal Form (CNF) known as CNF-SAT has been of central importance (Schoning and Toran 2013). These problems (known broadly as Boolean SAT problems) are concerned with deciding consistency (or existence of solutions) of Boolean equations in several variables. XOR-SAT is one special case of Boolean SAT where each equation is an exclusive OR (XOR) combination of variables.

Such linear XOR systems naturally appear in problems such as quadratic sieve method for prime factorization of numbers (Das 2016). Also in decoding of linear error correction coding, linear XOR systems

SpringSim-HPC 2017, April 23-26, Virginia Beach, VA, USA

need to be solved while optimizing the weight of solutions. Problems of finding all solution assignments with minimum Hamming weight, with maximum weight and of fixed weight are of different nature than the traditional problems of deciding satisfiability. All of these problems are addressed if the approach is aimed at finding all satisfying solutions. While solving general non linear polynomial equation systems, the XOR linear systems in terms of monomials can be solved as an intermediate problem (this is known as the XL approach to solving multivariate systems (Bard 2009)). Solving XOR Boolean systems stands out as a problem on its own and performances of algorithms aimed at solving such problems are relevant for understanding performances of solvers for general problems (Sule 2014). Although the XOR linear system problem is known to be of class P (as compared 3-CNF SAT of class NP complete), search for algorithms which scale to solving XOR problem of large sizes is as much important as scalability and performance of solvers for NP complete problems arising in practice by parallel algorithms.

The aim of this paper is to develop an approach for solving XOR linear systems over the Boolean algebra B_0 with a view to address following two objectives. This approach is based on implicant computation of Boolean formulas recently announced in (Sule 2016). In this paper we present the application of the ideas for the XOR linear case.

1. Finding all solutions of the system. This is not addressed by the known SAT approaches which are concerned with deciding satisfiability (or the existence of a solution). This problem of representing all solutions is of higher complexity than the satisfiability problem (Crama and Hammer 2011),(Desai and Sule 2014) in case of 2-CNF SAT problems. Moreover in applications such as Cryptography or Biological networks, satisfiability (or consistency) of the system is already known and it is required to find all solutions of the system. We shall follow the approach to represent all solutions in terms of implicants of equations as proposed in (Sule 2016).
2. Developing an approach which has inherent parallelism and can scaleup for solving large size problems over large number of processors. An important unresolved issue with solving Boolean equations is developing a solver which can scaleup with good efficiency for solving large systems arising in applications by parallel computation. Scalability of parallel solvers also depends on the algorithm and is affected by the number of processors. Our approach gives a method of computation in multiple parallel threads and is expected to have good scalability even over large number of processors. In fact its parallel performance improves with increased parallel resource.

Most systems arising from real life applications are sparse, i.e. each clause has only a small fraction of the complete set (large number) of variables. Such equations may have randomly distributed variables in each equation or in certain situation such as factorization of numbers dominant variables for small prime factors. While only local variables may be present in an equation when the variables have space dependent features. Gaussian elimination based algorithms cause loss of sparsity as computation progresses. On the other hand the implicant based approch increases sparsity due to substitutions.

Performances of other algorithms such as Grobner basis algorithm concerning scalability leave much desired as pointed out in (Bard 2009, Sule 2013). A general survey of parallel SAT solvers (Hammadi and Wintersteiger 2012) discusses many issues of scalability which are yet to be resolved. A limitation of SAT solvers is also that these are mainly designed for deciding satisfiability of CNF formulas and unless general systems are transformed to this form these methods are not applicable. The problem of solving Boolean equations is of considerable interest to Biological regulatory networks and is being studied from both theoretical and applied angle (Alan Veliz-Cuba and Laubenbacher 2014, Zou 2014). These references show that this problem is of current interest and hence it is important to continue search of new methods for solving Boolean equations which can scale up over large sizes of systems as well as large number of parallel processing elements. In short it is desirable to develop solvers for Boolean systems which provide inherent

parallelism in computation. In this paper we propose such a method for solving such problems associated with XOR systems.

1.1 Notations and background

The Boolean algebra referred in this paper is the two element algebra $B_0 = \{0,1,+,.,'\}$ with binary operations $+$, . denoting the well known *OR* (disjunction) and *AND* (conjunction) while $'$ denotes the complement operation. The Boolean ring $\{0,1,\oplus,.\}$ with $\oplus$ denoting the well known *XOR* shall also be denoted by B_0. The Boolean ring under $\oplus$ is equivalent to the binary field $GF(2)$. Two element Boolean algebra or ring are very well known and apart from the change of notation for operations we shall refer (Brown 2003) for their theory. Boolean functions $f : B_0^n \to B_0$ of n variables (denoted X) are equivalence classes of formal conjunctions and disjunctions of n-variables $x_i, i = 1,\ldots n$ and their complements x_i'. Such formal expressions when evaluated by assigning values of arguments from B_0 define Boolean functions. Boolean functions themselves form a Boolean algebra denoted $B_0(n)$. For a Boolean function f the set of all *satisfying assignments* is the set of points a in B_0^n such that $f(a) = 1$. This set is denoted by $S(f)$. A term in X is a function

$$t(X) = \prod_{1 \le i \le n} x_i^{\alpha_i} \ \alpha_i \in \{0,1\}$$

where for a variable x, $x^\alpha = x$ when $\alpha = 1$ and $x^\alpha = x'$ when $\alpha = 0$. The set of indices i in a term t shall be called its *support* and denoted $\sup(t)$. Clearly

$$S(t) = \{x_i = \alpha_i \forall i \in \sup(t), x_i = D \forall i \ni \sup(t)\}$$

where D denotes an arbitrary assignment. Hence we represent the set $S(t)$ by the compact notation (t) which denotes the partial assignments for $x_i, i \in \sup(t)$ in $S(t)$.

1.2 Implicants and representation of set of all satisfying assignments

An implicant of a Boolean function $f(X)$ is a term $t(X)$ such that $t \le f$ in the Boolean algebra of functions $B_0(n)$. The substitution of partial assignments (t) in f is denoted as f/t and is known as the *ratio* or *cofactor* of f by t. We observe the obvious result,

Proposition 1. If $f(X)$ is a Boolean function and t a term in X then following statements follow the implication $1) \Rightarrow 2) \Rightarrow 3) \Rightarrow 1)$.

1. t is an implicant of f.
2. $f/t = 1$.
3. $(t) \subset S(f)$.

A set of implicants $I(f)$ of f is said to be *complete* if $f(a) = 1$ for some a then there exists a t in $I(f)$ such that $t(a) = 1$. Hence when $I(f)$ is complete for f we have the equivalent expressions as given in the following,

Proposition 2. Following statements are equivalent

1. $I(f)$ is a complete set of implicants.
2. $S(f) = \bigcup_{t \in I(f)} (t)$
3. $f = \sum_{t \in I(f)} t$

where the sum in the third expression is an OR sum of implicants t in I.

Proof. 1) $\Leftrightarrow$ 2). The inclusion $\bigcup_{t\in I(f)}(t) \subset S(f)$ follows from the definition of implicant. Conversely, let $a \in S(f)$, then since $I(f)$ is complete there exists a $t \in I(f)$ such that $t(a) = 1$. Hence

$$S(f) \subset \bigcup_{t\in I}(t)$$

1) $\Leftrightarrow$ 3). By definition

$$\sum_{t\in I(f)} t \leq f$$

But since $I(f)$ is complete, if $f(a) = 1$ there is an implicant t such that $t(a) = 1$. Hence

$$\sum_{t\in I(f)} t(a) = f(a)$$

from which the equivalence follows. □

Above proposition is a basis of our algorithm for computing all solutions of XOR systems.

1.3 XOR-SAT and associated problems

The linear XOR-SAT problem without constraints on solutions, in n variables over B_0 is defined by a system of m equations of the form

$$\bigoplus_{j=1}^{n} a_{ij}x_j = b_i, i = 1,\ldots,m \tag{1}$$

where a_{ij}, b_i are elements of B_0. The problem is to find all n-tuples $a = (a_1,\ldots,a_n)$ in B_0^n such that each a gives a solution assignment $x_i = a_i$. The basic XOR-SAT problem we consider is 1) to find all assignments for $X = \{x_1,\ldots,x_n\}$ in B_0^n which satisfy equations (1). Such assignments when they exist (i.e. when the system is satisfiable) are finite in number. There are important associated problems. If $w : B_0^n \to \mathbb{W}$ is a non-negative integer valued function (representing weight of an assignment) then we have the associated problems 2) to find all solutions a of the system such that $w(a) < q$ where q is a specified non-negative number and 3) to find all solutions a of the system such that $w(a)$ is minimum. Clearly if we solve the problem 1) then the associated problems are solvable by search over the solution set. Our approach to represent the solutions in terms of implicants makes such a search feasible.

One of the central issues with these problems is that although the number of solutions are always finite, the number grows exponentially in the number of free assignments of variables in each solution. Hence solutions of the above problems need to be compactly represented rather than just enumerated. In fact simply enumerating the finite set of assignments is not practically feasible in large sized problems. Clearly the best way to represent such solution sets are by collecting the terms corresponding to variables which have fixed assignments since variables with free assignments need not be explicitly shown in a solution. This way the exponential number of solutions can be represented compactly by fixed variable assignments. The representation of $S(f)$ as in Proposition 2 provides such a compact representation for satisfying assignments of a Boolean function f. We extend this representation for the solution set of the systems of XOR linear equations. Next, the problem of scalability can also be addressed by this compact way to represent assignments. For instance the implicants representing any solution of the system must necessarily also be implicants for solving a single equation. Hence an algorithm which at each step restricts search of assignments over small

number of variables involved in a single equation and carries out independent search along parallel threads can provide scalability of computation. Our approach to solving the XOR linear problem is based on these ideas.

2 IMPLICANT BASED APPROACH FOR SOLUTION OF BOOLEAN SYSTEMS

Our approach to the problem of representing all solutions of the system (1) depends on constructing a complete set $I(E)$ of all implicants corresponding to an equation E of the system. Let E denote an equation in the linear system of the form

$$a_{j1}x_{j1} + a_{j2}x_{j2} + \ldots + a_{jk}x_{jk} = b_j$$

where $1 \leq j1 \leq j2 \leq \ldots \leq jk \leq n$. Consider a Boolean function $f(x_{ji})$ in variables x_{ji} whose set of satisfying assignments is

$$S(f) = \{x_{ji} \in B_0 | E \text{ is satisfied } \}$$

Then it follows that a complete set of implicants $I(f)$ characterizes the set of all satisfying assignments for the equation E. We shall thus denote this set of implicants as $I(E)$ and call it a *complete set of implicants* of equation E and call this function as the *true value function* of equation E. Thus the set $I(E)$ represents all solutions of the equation E denoted as $S(E)$. Then for each t in $I(E)$, (t) denotes a set of partial assignments satisfying E, hence

$$S(E) = \{ \bigcup_{t \in I(E)} (t) \}$$

this is then a compact way to represent all solutions (or the satisfying assignments) of an equation E.

2.1 Satisfying assignments of simultaneous equations

Now consider the problem of representing satisfying assignments of two simultaneous equations in variables X. So let E_1 and E_2 be these two equations. If we compute a complete set of implicants $I(E_1)$ then the two simultaneous equations are consistent iff for all t in $I(E_1)$ the substitution E_2/t equivalently f_2/t does not result into a contradiction $f_2/t = 0$ (or E_2 not satisfied). We can formally write the following.

Proposition 3. Two simultaneous equations E_1 and E_2 are consistent iff $f_2/t \neq 0$ (i.e. E_2 is satisfied) for some t in a complete set of implicants $I(E_1)$. A complete set of implicants $I(E_1, E_2)$ of the simultaneous system of equations is given by either of the sets

$$\begin{aligned} I(E_1, E_2) &= \bigcup_{t \in I(E_1), s \in I(E_2/t)} \{ts\} \\ &= \bigcup_{s \in I(E_2), t \in I(E_1/s)} \{ts\} \end{aligned}$$

Proof. Let $I(E_1)$ be a complete set of implicants of equation E_1. Then each t in $I(E_1)$ represents partial assignment (t) in the set of all solutions of E_1. Hence the simultaneous equations have a solution iff there is a partial assignment (t) whose satisfying set $S(t)$ intersects the solution set of E_2. This is true iff when (t) is substituted in E_2 does not lead to contradiction. This is the condition $f_2/t \neq 0$. When there is no contradiction the resulting equation is E_2/t. If this equation has a satisfying partial assignment s in the remaining variables then ts is a simultaneous implicant of both equations representing a partial assignment $(t)(s)$. Taking union over all implicants t in $I(E_1)$ for which E/t is not a contradiction thus gives the formula $I(E_1, E_2)$. Since the order of equations leaves the solutions invariant the formula is symmetric. □

This proposition is the basis of our algorithm to compactly represent all satisfying assignments of simultaneous equations. We first consider few examples.

2.2 Examples of representing solutions by implicants

Example 1. Consider the two equation system

$$\begin{aligned} w \oplus x \oplus y &= 1 \\ x \oplus y \oplus z &= 0 \end{aligned}$$

For first equation E_1 we have

$$I(E_1) = \{wx'y', w'xy', w'x'y, wxy\}$$

This gives

$$\begin{aligned} E_2/wx'y' &\Leftrightarrow \{z = 0\} \\ E_2/w'xy' &\Leftrightarrow \{z = 1\} \\ E_2/w'x'y &\Leftrightarrow \{z = 1\} \\ E_2/wxy &\Leftrightarrow \{z = 0\} \end{aligned}$$

Since no contradiction took place the system is consistent and every implicant of E_1 leads to a solution. Hence we have

$$\begin{aligned} I(E_1, E_2) &= \{wx'y'z', w'xy'z, \\ &\quad w'x'yz, wxyz'\} \end{aligned}$$

which gives the set of all solution assignments satisfying the simultaneous equations.

Consider another example of a consistent system.

Example 2.

$$\begin{aligned} w \oplus x \oplus y &= 1 \\ w \oplus y &= 1 \end{aligned}$$

The first equations is same as above hence has same $I(E_1)$. We compute the substitutions in the second equation

$$\begin{aligned} E_2/wx'y' &\Leftrightarrow \{1 = 1\} \\ E_2/w'xy' &\Leftrightarrow \{0 = 1\} \\ E_2/w'x'y &\Leftrightarrow \{1 = 1\} \\ E_2/wxy &\Leftrightarrow \{0 = 1\} \end{aligned}$$

Two of these are contradictions. For the other two $E_2/t = 1$ hence t is an implicant of E_2 also. Hence we have

$$I(E_1, E_2) = \{wx'y', w'x'y\}$$

2.3 Notation for larger examples

We shall now introduce a notation to represent larger systems and implicants. Consider the system of linear equations.

$$\begin{aligned} x_3 \oplus x_5 \oplus x_7 &= 1 \\ x_1 \oplus x_4 \oplus x_5 &= 0 \\ x_2 \oplus x_5 \oplus x_6 &= 1 \\ x_3 \oplus x_7 &= 0 \\ x_5 \oplus x_7 &= 0 \end{aligned}$$

We represent the above system denoted S as the set

$$\{[3,5,7,0,1], [1,4,5,0,0], [2,5,6,0,1], [3,7,0,0], [5,7,0,0]\}$$

(This notation is inspired by notation for CNF clauses well known in DIMACS notation but is not the same). We also denote an implicant term of the type $t = x'_3 x_5 x'_6 x_7$ as well the partial assignment (t) by the notation $(-3,5,-6,7)$. Also the partial assignment of a product of two implicants ts is denoted as $(t)(s)$. In the above system S, consider first an equation with minimum number of variables

$$[3,7,0,0]$$

A complete implicant set for this equation is

$$\{(-3,-7),(3,7)\}$$

Let S/t denote the system obtained by substitution of assignment $t = 1$ in all equations of S. Then (equations which are trivially satisfied and result in tautologies $0 = 0$ or $1 = 1$ are dropped from the new system)

$$S/(-3,-7) =$$
$$\{[5,0,1],[1,4,5,0,1],[2,5,6,0,1],[5,0,0]\}$$
$$S/(3,7) =$$
$$\{[5,0,1],[1,4,5,0,1],[2,5,6,0,1],[5,0,1]\}$$

In both the resulting reduced systems we have implicant of first equation as $\{(5)\}$. Substitution of this implicant gives rise to two new reduced systems

$$\{[1,4,0,0],[2,6,0,0],[1,0,0]\}$$
$$\{[1,4,0,0],[2,6,0,0],[1,0,1]\}$$

The first system has an inconsistent equation $1 = 0$ while the second system has no contradiction. Hence the complete set of satisfying assignments are represented by the implicant set by taking the product of the previous implicant with that of the second system

$$\{(3,7)(5)(1,4),(3,7)(5)(-1,-4),$$
$$(3,7)(5)(2,6),(3,7)(5)(-2,-6)\}$$

3 PROPOSED ALGORITHM TO FIND ALL SOLUTIONS

We now propose the algorithm for representing all solutions of a system of equations S in terms of an implicant set or return an empty set if the system is inconsistent. The algorithm starts with selecting an equation E called a pivot equation. (A suitable choice of a pivot is an equation with minimum number variables). A complete set of implicants denoted $I(E)$ is then computed. All satisfying assignments of E are represented by all partial assignments satisfying these implicants. The system of equations is then reduced to S/t for a selected implicant t. The processes is repeated until a contradiction is reached when an equation in the reduced system is contradicted when evaluated at an implicant or else an augmented implicant set is returned. The final sets of implicants returned in each thread determine the partial assignments of all satisfying assignments of the system. This is described in the following pseudocode of the algorithm 1.

4 TIME COMPLEXITY AND RESULTS ON RANDOM CASES

Performance of the above XOR system solver has been evaluated on systems $Ax = b$ where matrices A are over the Boolean ring B_0 and nonsingular of size n. The vector x is an n tuple of variables to be solved and b is a known n tuple. The operation Ax uses the Boolean ring operations of product and XOR sum. Systems of different sizes varying from $n = 40$ to $n = 500$ are selelcted in two different sets in which A is chosen non-singular. Such random non-singular matrices are chosen by randomly transforming a matrix in Hessenberg form as described below while b is a random vector. All these systems have a unique solution since A is non singular by design.

Algorithm 1: All Solutions: (Reduction of a system by Implicants)

Input : The linear system of equations $S : f_k(X) = b_k$ (k-th equation denoted as E_k), $k = 1,2,\ldots n$, in variables $X = \{x_1,\ldots,x_n\}$.

Output: set of all solution assignments.

Choose an equation E in S with least number of variables;
Compute a complete set of implicants $I(E)$ of E and order it as $\{t_1,\ldots,t_{N_k}\}$;
Select lowest order implicant t in $I(E)$;
Start thread:
Reduce the system: pick equations from S of indices j and compute f_j/t.
if $f_j/t = 0$ *for some* j **then**
    End thread;
    Discard the implicant;
    Select next implicant in the order in 3 and restart thread;
else
    Compute the reduced equations f_j/t for all j;
    Denote the reduced system as S/t;
    Delete the equations for j such that $f_j/t = 1$;
end
if *all* $f_j/t = 1$ **then**
    Return $(I)(t)$;
else
    set $S \leftarrow S/t$ $I \leftarrow (I)(t)$.
end
Repeat 6 with this I untill $S = \emptyset$;
`/* The set` I `is the set of implicants denoting partial assignment of solutions of` S `in the current thread started by the implicant. */`
Collect union of partial assignments given by I in all threads;
This gives the set of all solution assignments;

4.1 Procedure for generating matrix A

Matrix A is generated by making random elementary transformations on a non singular Boolean matrix in Hessenberg form. For instance a 5×5 non-singular matrix in Hessenberg form is

$$H = \begin{bmatrix} * & 1 & 0 & 0 & 0 \\ * & * & 1 & 0 & 0 \\ * & * & * & 1 & 0 \\ * & * & * & * & 1 \\ 1 & * & * & * & * \end{bmatrix}$$

where $*$ denotes an arbitrary 0 or 1 entry. Similar construction is considered to generate a general non-singular matrix in Hessenberg form. Then the A matrix is obtained by random elementary row operations on H. This procedure is carried out in following steps.

1. set counter $cnt = 0$.
2. Randomly generate i from the range 1 to n.
3. Randomly generate j from the range 1 to n.
4. If $i \neq j$ then
   (a) add elements of row i to row j i.e.
   $Row(j) = Row(j) + Row(i)$. Elementary addition are over binary field $\mathbb{F}_2$.
   (b) make $cnt = cnt + 1$
5. If $i = j$ then Go To step 2.
6. steps 2 to 5 were followed until $cnt \geq (n * 10/100)$

4.1.1 Analysis of complexity of solving the system and experimental test cases

Algorithm 1 essentially involves two computational steps at every stage when a pivot equation is chosen. These are as follows.

1. Generation of a complete set of implicants $I(E)$ for the pivot equation.
2. Substitution of the partial assignment (t) of an implicant $t \in I(E)$ in the rest of the equations to get reduced system S/t.

These two steps define a thread segment. The starting point of this thread is an implicant t which is to be used for reduction of the system. At the end of the thread implicant t is either discarded due to a contradiction or is qualified as an implicant of the system or leaves a reduced system. All these thread segments are independent computations. Hence the reduction of the original system progresses along parallel threads. If we assume that there is sufficient parallel resource available then at each stage of start of a thread segment the computations can be carried out on independent processors. Hence the time required for solving the XOR system with sufficient parallel resource is equal to the time required for the thread requiring longest time consisting of a sequence of thread segments. We assume that the time for implicant computation in each thread segment is constant (this is justified for sparse systems in which number of nonzero entries in any equation is small and bounded from above). Similarly the time taken for substitution of a partial assignment in equations is also considered constant (upper bound). In each thread segment at least one variable assignment is discovered. Hence a longest thread has atmost n steps. Hence the time taken for longest thread is of order $O(n)$ asssuming all thread segments are exeuted independenly (or in parallel).

4.2 Experimental test case results

Experimental cases of computation with this algorithm are documented in the following table.In these random samples of systems, for each n, 10 binary random matrices A of size $n \times n$ and 10 n-tuple vectors b of were selected. The systems $Ax = b$ were solved sequentially by the above algorithm and time taken for each thread segment were measured. From these records of thread segment time, the time required for solving the system when all threads could be executed in parallel was calculated. This is the time taken for the sequence of thread segments which takes the longest time, while the time required to solve each system sequentially is the sum of times taken for all thread segments. From these measurements the averages of time taken for the longest threads and total were calculated for the random samples. These are plotted in the following table shown. The graph shows approximately linear $O(n)$ trend as expected. The slight trend visible of the type $O(n^{1+\alpha})$ in the average time for longest threads is due to the fact that the experimental test cases had nonzero terms in any equation equal to half of n. Hence the time taken for computing implicant sets in the thread segments was not constant as assumed but was actually weakly $O(n^{\alpha})$ for $\alpha < 1$ with a small constant. Hence the longest thread time appears to tend towards $O(n^{1+\alpha})$.

As described above our computations are completely sequential. However due to the independence of thread segments through which the algorithm works it is theoretically possible to make estimates of parallel efficiency. If it is assumed that an infinite parallel resource is available for implementing the parallel threads, then the maximum speedup possible is the ratio

$$\begin{aligned} \text{speedup} &= \frac{\text{Time Taken for Sequential Solution}}{\text{Time Taken for longest Thread}} \\ &= \frac{\text{Sum of Time taken for all threads}}{\text{Time taken for the longest thread}} \end{aligned}$$

The table also document this maximum speedup in terms of average time taken for longest thread and the sequential solution. The table also show that speedup is higher if number of threads are larger. Although this means larger requirement of memory and parallel processors, this also gives a positive indication that this algorithm is expected to be scalable for large data and large number of processors.

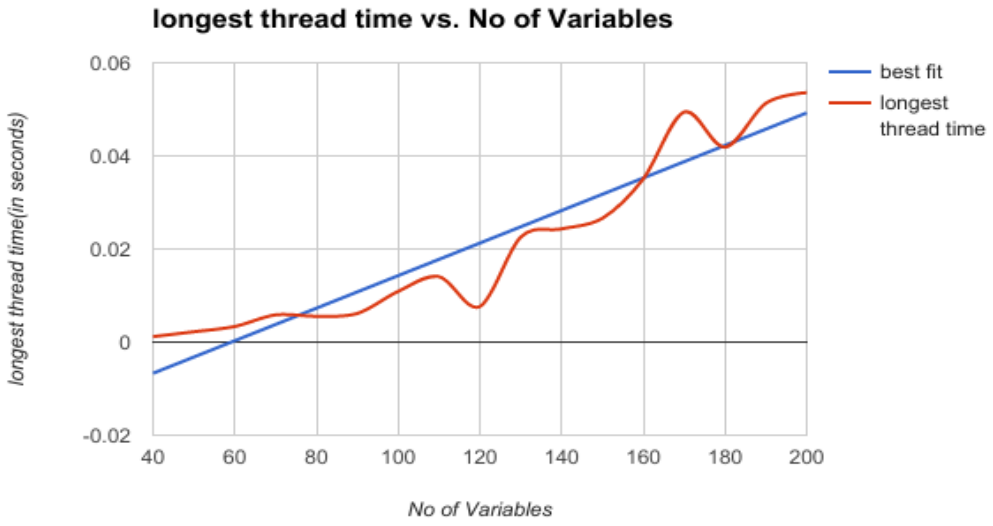

Figure 1: Longest Thread time Vs No of Variables.

Table 1: Experimental Results for all solutions

Number of Variables	Average Number of Threads	Average Time Taken for longest Thread	Average Time Taken for Sequential Solution	Maximum speedup predicted by parallel computation
40	5	1.18E-03	5.64E-03	8
50	9	2.22E-03	1.64E-02	8
60	9	3.35E-03	2.41E-02	11
70	13	5.85E-03	5.83E-02	11
80	15	5.55E-03	9.16E-02	17
90	30	6.14E-03	1.85E-01	35
100	19	1.09E-02	1.74E-01	22
110	51	1.41E-02	3.76E-01	53
120	41	7.64E-03	3.86E-01	76
130	122	2.26E-02	1.57E+00	150
140	233	2.43E-02	2.96E+00	137
150	153	2.66E-02	2.82E+00	130
160	176	3.51E-02	3.72E+00	139
170	203	4.94E-02	7.89E+00	152
180	190	4.19E-02	6.02E+00	160
190	393	5.13E-02	1.18E+01	230
200	274	5.36E-02	1.87E+01	350

5 CONCLUSION

A parallel solver for special Boolean systems called XOR linear systems is presented in this paper which represents all solutions of the system. Such systems are traditionally solved by Gaussian elimination over the binary fields or Lanczos methods. The proposed solver treats these as Boolean systems and all operations performed are Boolean. The algorithm splits the computation in terms of independent threads and hence when sufficient parallel resource is available, the algorithm gives an idea of the maximum speedup achievable by parallel computation. Parallel performance speed ups are tabulated for random systems and show promising speed ups. Although systems required to be solved in applications are non linear, the linear system speed up performance is relevant since the basic of operation of substitution of assignments does not get affected by linear nature of functions. Hence the results of this paper are of interest even for general non linear problems for solving Boolean systems.

REFERENCES

Alan Veliz-Cuba, Boris Aguilar, F. H., and R. Laubenbacher. 2014. *Steady state analysis of Boolean molecular network models via model reduction and computational algebra*. 221 ed. BMC Bioinformatics.

Bard, G. 2009. *Algebric Cryptanalysis*. Springer.

Bolouri, H. 2008. *Computational modeling of gene regulatory networks.* Imperial College Press.

Brown, F. M. 2003. *Boolean analysis: The logic of Boolean equations*. Dover.

Crama, Y., and P. Hammer. 2011. *Boolean functions. Theory, algorithms and applications Encyclopedia of Mathematics and its applications*, Volume 142. Cambridge.

Das, A. 2016. *Computational number theory*. CRC Press.Prentice-Hall, Inc.

Desai, M. P., and V. Sule. 2014. "Generalized cofactors and decomposition of Boolean satisfiability problems". *arXiv.org/cs.DS/1412.2341v1*.

Hammadi, Y., and C. M. Wintersteiger. 2012. "Seven Challenges in Parallel SAT Solving". *Challenge paper AAAI 2012 Sub-Area spotlights track. Association of Advancement of Artificial Intelligence.*.

Schoning, U., and J. Toran. 2013. *The satisfiability problem, algorithms and analyses*. Lehmanns media.

Sule, V. 2013. "Generalization of Boole-Shannon expansion, consistency of Boolean equations and elimination by orthonormal expansion". *arXiv.org/cs.CC/1306.2484v3*.

Sule, V. 2014. "An algorithm for Boolean satisfiability based on generalized orthonormal expansion". *arXiv.org/cs.DS/1406.4712v3*.

Sule, V. 2016. "Implicant based parallel all solution solver for Boolean satisfiability". *arxiv.org/cs.Ds/1611.09590v3*.

Zou, Y. M. 2014. "An algorithm for detecting fixed points of Boolean networks". *arXiv.org:1404.5515v1[q-bio.QM]*.

AUTHOR BIOGRAPHIES

JAYASHREE V. KATTI is an Associate Professor at Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University. She is currently pursuing PhD in Computer Science and Engineering. Her research interests are Cryptography, Algebraic Cryptanalysis, Visual Cryptography, High performance computing. Her email address is jayashree.katti@gmail.com.

VIRENDRA SULE has been teaching at Indian Institutes of Technology for over 25 years and has published research in Control and Systems theory. Since recent times he has been working in Cryptography and computational algorithms for Boolean systems. For a brief period he worked with the TATA startup venture Computational Research Laboratories on parallel computing. His email address is viren.sule@gmail.com.

B.K.LANDE currently working as Professor in Narsee Monjee Institute of Management Studies, Mumbai. His research interests are Controls and Communication Engineering. His email address is bklande@gmail.com.